ストリーム処理DSLについてのサーベイ

インターンシップ報告会

水野 雅之

2018年3月23日

モチベーション:IoT デバイスの開発

- センサーからデータが時事刻々と流入
- リアルタイムな処理が必要
- 計算機資源に限りがある

⇓

低レイヤも制御できるストリーム処理 DSL が欲しい

当初の実装:Haiyu.hs

- Actor モデル
  - 各コンビネータは非同期に並列動作
- Arrow で組み立てる

しかし

- 想定される処理は恐らく GPU 律速
  - センシングだけ速くても，処理待ちのデータが溜まるだけ

⇒ 各コンビネータは同期しても良いのでは

埋め込み DSL

既存のホスト言語上に DSL を構築する手法

長所:

- **ホスト言語の機能を流用可能**
  - **構文解析**
  - **型チェック**

短所:

- **ホスト言語の表現力に制約される**
  - **ホスト言語が型安全でなければ，型安全な DSL は作りづらい**

Haskellのストリーム処理ライブラリ

- **代表例**:Conduit，Pipes，io-streams
- **遅延IOを御するために生まれた**
  - **例外**
  - **リソースの管理**
- **入出力を含んだコード片も部品として組み合わせやすくなる**
- **IO等のモナドを合成する設計**

検討したDSL

- C言語のコードを出力する，Haskellの埋め込みDSL
  - メタプログラミングを意識させない
  - Parametricity（seq等を除く）
  - 幽霊型
- コード生成を行うモナドだけ実装し，ストリーム処理ライブラリで包む
  - 好みにより取り替え可能に

テスト実装（コード生成を行うモナド）

- とりあえず算術式に限定して実装
- データフローグラフを強連結成分分解すればCコードが得られる

問題点

- コード生成の都合上，ループや
  シェアリングを明示する必要がある
  - ストリーム処理ライブラリと組み合
    わせる障害に
- 幽霊型を入れようとすると一癖ある

“*Je nái pas le temps.*” — Évariste Galois